User Guide

Front Panel Designer

Reference:	TLT-0448-MAN-Front-Panel-Designer
Product Version:	--
Revision:	A

Table of Contents

1. Introduction 5
    1.1. Description 5
    1.2. Overview of Process 5
    1.3. Dependencies 5
2. The Front Panel Project 6
    2.1. Creating a Front Panel Project 6
    2.2. Project Content 6
3. Front Panel Definition Files (.fp files) 8
    3.1. File Content 8
    3.2. Working with fp Files 8
4. Displays 10
5. Inputs 11
    5.1. Listeners 11
    5.2. Testing Events 11
6. Export 12
    6.1. Exporting the Front Panel 12
    6.2. Configuring the MicroEJ Framework 12
7. Designer Mode 13
8. Document History 14

List of Figures

2.1. New Front Panel Project Wizard 6
2.2. Project Content 6
3.1. Front Panel XML Schema 8
3.2. Working Layout Example 9
4.1. Front Panel XML File Snippet: Display 10
5.1. Front Panel XML File Snippet: Push Button 11
6.1. Exporting the Front Panel 12

1 Introduction

1.1 Description

A major strength of the MicroEJ® environment is that it allows applications to be developed and tested in a simulator rather than on the target device, which may not yet be built. To make this possible for devices that have a display or controls operated by the user, such as a touch screen or buttons, the simulator must connect to a “mock” of the control panel (the “front panel”) of the device. This mock is called the *mockFP*. The mockFP generates a graphical representation of the required front panel, and is displayed in a window on the user's development machine when the application is executed in the simulator. The mockFP is the equivalent of the five embedded stacks (alpha-numeric, audio, display, input and leds) of the EmbJPF (see the MicroUI fragment chapter in the UI Pack User's Manual).

1.2 Overview of Process

To create a mockFP the user:

1. Creates a new Front Panel project and associates it with a work in progress JPF.
2. Creates an image of the required front panel. This could be a photograph or a drawing.
3. Defines the content and layout of the front panel by editing an XML file (called an `fp` file).
4. Creates images to animate the operation of the controls (e.g. button down image).
5. Creates `Listeners` that generate the same MicroUI input events as the hardware.
6. Creates a `DisplayExtension` that configures the simulated display to match the real display.
7. Previews the front panel to check the layout of controls and the events they create, etc.
8. Exports the Front Panel project into the work-in-progress JPF.

Note: this document contains some background information not essential for most users – this information is shown in separate boxes headed “Behind the scenes...”.

1.3 Dependencies

The front panel extension requires the fragments `microui` and `frontpanel` to have been installed in the work-in-progress JPF. Please refer to the microui and frontpanel fragment chapters of the UI Pack User's Manual.

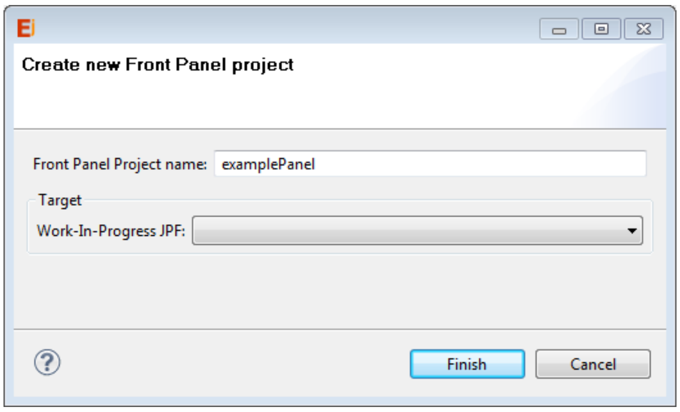
2 The Front Panel Project

2.1 Creating a Front Panel Project

A Front Panel project is created using the New Front Panel Project wizard. Select

New → Project... → MicroEJ → Front Panel Project

The wizard will appear:



Create new Front Panel project

Front Panel Project name: examplePanel

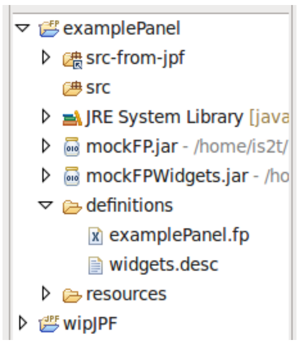
Target

Work-In-Progress JPF:

Figure 2.1. New Front Panel Project Wizard

Enter the name for the new project and select the work-in-progress JPF with which this project should be associated. Note that this association can be changed using the **Associated JPF** page of the project's properties.

2.2 Project Content



```
examplePanel
  src-from-jpf
  src
  JRE System Library [java
  mockFP.jar - /home/is2t/
  mockFPWidgets.jar - /ho
  definitions
    examplePanel.fp
    widgets.desc
  resources
wipJPF
```

Figure 2.2. Project Content

Note that the project icon has the letters “FP” in the top right corner.

A Front Panel project has the following structure and content:

- The `src-from-jpf` folder is linked to the work-in-progress JPF and contains an interface `com.is2t.microej.microui.Constants` which defines constants mapping event generator names to numeric values. These constants are used when defining `Listeners`, and match the MicroUI definition for the work-in-progress JPF. This interface must not be edited. See below for more details about defining listeners.

- The `src` folder is provided for the definition of `Listeners` and `DisplayExtensions`. It is initially empty. The creation of `Listeners` and `DisplayExtensions` will be explained later.

- The `JRE System Library` is referenced, because a Front Panel project needs to support the writing of Java for the `Listeners` and `DisplayExtensions`.

- The `mockFPWidgets.jar` references the work-in-progress JPF and contains the code for the front panel simulation, the widgets it supports and the types needed to implement `Listeners` and `DisplayExtensions`.
- The `definitions` folder holds the file or files that define the content and layout of the front panel, with a `.fp` extension (the fp file or files), plus some supporting files. A newly created project will have a single fp file with the same name as the project, as shown above. The content of fp files is explained later in this document.
- The `widgets.desc` file contains descriptions of the widgets supplied with the XPF. It is used by the Front Panel Designer tool and must not be edited.
- The `resources` folder holds images used to create the mockFP. It is initially empty.

3 Front Panel Definition Files (.fp files)

3.1 *File Content*

An fp file is an XML file with the following structure:

```
<?xml version="1.0"?>
<frontpanel
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xml.is2t.com/ns/1.0/frontpanel"
    xsi:schemaLocation="http://xml.is2t.com/ns/1.0/frontpanel.fp1.0.xsd">

    <description file="widgets.desc"/>

    <device name="example" skin="example-device.png">
    <body>
        <[widget-type] id="0" x="54" y="117" [widget-attributes] />
        <[widget-type] id="1" x="266" y="115" [widget-attributes] />
        ...
    </body>
    </device>
</frontpanel>
```

Figure 3.1. Front Panel XML Schema

The `description` element must appear exactly as shown. It refers to the `widgets.desc` file mentioned above.

The `device skin` must refer to a `png` file in the `resources` folder. This image is used to render the background of the front panel. The widgets are drawn on top of this background.

The `body` element contains the elements that define the widgets that make up the front panel. The name of the widget element defines the type of widget. The set of valid types is determined by the work-in-progress JPF associated with the project. Every widget element defines an `id`, which must be unique for widgets of this type, and the `x` and `y` coordinates of the position of the widget within the front panel (0,0 is top left). There may be other attributes depending on the type of the widget.

Refer to the UI Pack Reference Manual for more information about the content of the fp file.

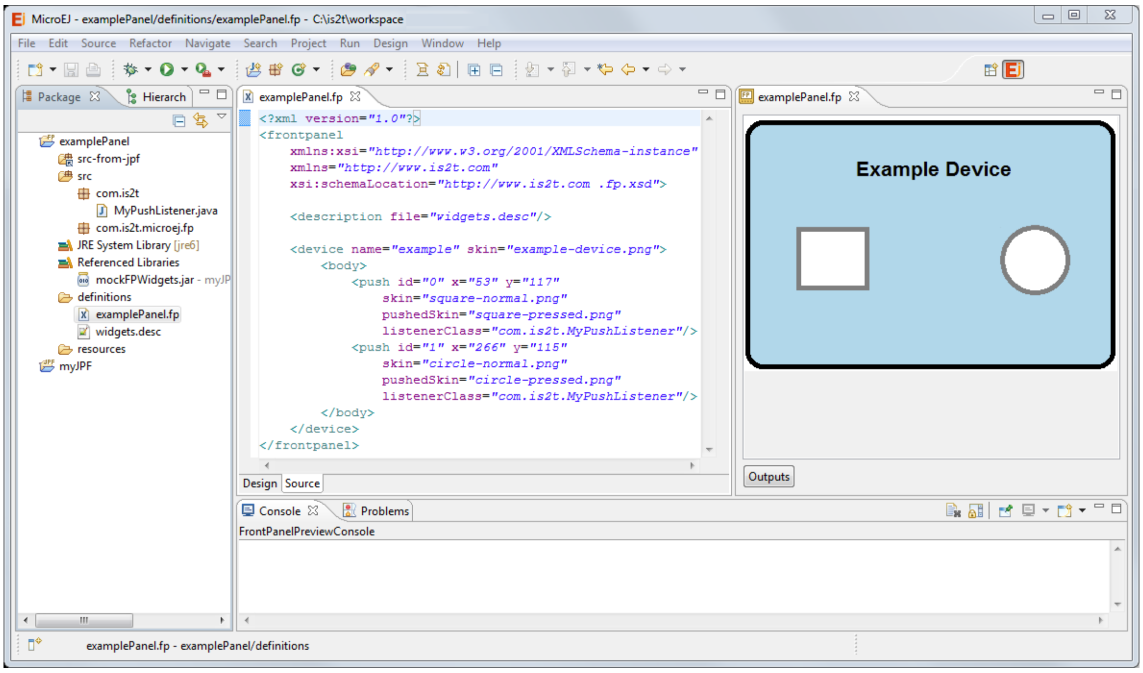
3.2 *Working with fp Files*

To edit an fp file, open it using the Eclipse XML editor (right-click on the fp file, select **Open With** → **XML Editor**). This editor features syntax highlighting and checking, and content-assist based on the schema (XSD file) referenced in the fp file. This schema is a hidden file within the project's definitions folder. An incremental builder checks the content of the fp file each time it is saved and highlights problems in the **Eclipse Problems** view, and with markers on the fp file itself.

A preview of the front panel can be obtained by opening the Front Panel Preview (either **Window** → **Show View** → **Other...** → **MicroEJ** → **Front Panel Preview** or **Window** → **Open Perspective** → **Other...** → **MicroEJ** followed by **Window** → **Reset Perspective...**).

The preview updates each time the fp file is saved.

A typical working layout is shown below.



```
<?xml version="1.0"?>
<frontpanel
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.is2t.com"
    xsi:schemaLocation="http://www.is2t.com .fp.xsd">

    <description file="widgets.desc"/>

    <device name="example" skin="example-device.png">
        <body>
            <push id="0" x="53" y="117"
                skin="square-normal.png"
                pushedSkin="square-pressed.png"
                listenerClass="com.is2t.MyPushListener"/>
            <push id="1" x="266" y="115"
                skin="circle-normal.png"
                pushedSkin="circle-pressed.png"
                listenerClass="com.is2t.MyPushListener"/>
        </body>
    </device>
</frontpanel>
```

Figure 3.2. Working Layout Example

Within the XML editor, content-assist is obtained by pressing **ctrl+space**. The editor will list all the elements valid at the cursor position, and insert a template for the selected element.

4 Displays

In order to use a display in a Front Panel, a `Display` widget needs to be specified, along with a `DisplayExtension` which defines the characteristics of the display.

Here is an example snippet of an fp file for specifying a `Display` widget:

```
<pixelatedDisplay
        id="0"
        x="162" y="114"
        width="320" height="240"
        initialColor="0x000000"
        extensionClass="com.is2t.MyDisplayExtension"/>
```

Figure 4.1. Front Panel XML File Snippet: Display

The `extensionClass` attribute specifies the fully qualified class name of a class that extends `com.is2t.microej.frontpanel.display.DisplayExtension` or an appropriate subclass of it. This implementation needs to be in the `src` folder of the Front Panel Project.

Please refer to the UI Pack Reference Manual for more information about the display extension class.

5 Inputs

5.1 Listeners

Widgets in the simulated front panel that accept user input will generate events that must be the same as the events generated by the real hardware. For widgets that generate events, a `Listener` class must be specified within the fp file.

As an example, consider this snippet of an fp file for defining a push button:

```
<push id="0" x="54" y="117"
        skin="square-normal.png"
        pushedSkin="square-pressed.png"
        listenerClass="com.is2t.MyPushListener" />
```

Figure 5.1. Front Panel XML File Snippet: Push Button

The value of the `listenerClass` attribute is the fully qualified name of a class which has to implement the `com.is2t.microej.frontpanel.input.listener.PushButtonListener` interface. This class, `com.is2t.MyPushListener`, is written by the developer of the Front Panel, in the `src` folder of the Front Panel Project. It can reference the `PushButtonListener` interface because that is provided in the referenced library, `mockFPWidgets.jar`, which exists within the associated work-in-progress JPF.

Please refer to the UI Pack Reference Manual for more information about the inputs.

5.2 Testing Events

If a widget is selected using the mouse in the Front Panel Preview, information about the generated event is shown in the Eclipse console. The format of the information shown depends on the configuration of the work-in-progress JPF.

Behind the scenes...

The Front Panel Preview converts the raw int value of the event, as generated by the widget, into the information displayed in the console by calling the

```
com.is2t.inputs.EventDecoder.toString(int)
```

method. That class is provided in the mockFPWidgets.jar in the mocks folder of the work-in-progress JPF. However, the user can replace the standard behavior by implementing the class in the Front Panel project. If the previewer does not find the class or the required method it displays a warning in the console and shows just the raw value of events.

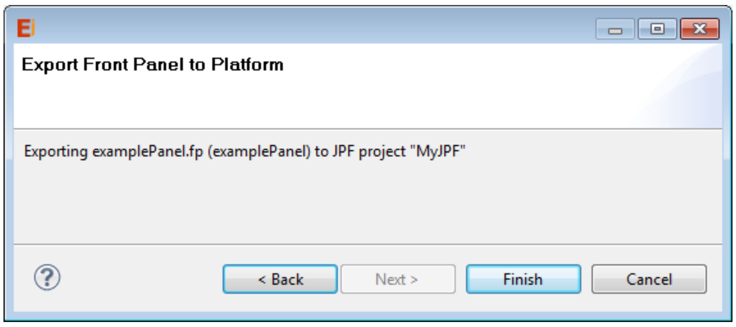
6 Export

Before the front panel can be used by applications it must be exported to the work-in-progress JPF.

6.1 *Exporting the Front Panel*

Select the fp file you want to export, and then

File → **Export...** → **MicroEJ** → **Front Panel**


Export Front Panel to Platform

Exporting examplePanel.fp (examplePanel) to JPF project "MyJPF"

Figure 6.1. Exporting the Front Panel

There are no further settings required. The project contents are exported to the associated work-in-progress JPF, including information about which fp file is to be used. If there is only one fp file defined in your project, then you can select the project rather than the fp file.

Behind the scenes...

The export wizard generates `mockFPWidgetsExtension.jar` in the `mocks` folder of the work-in-progress JPF. This jar contains all the contents of the project.

Note that it is not possible to export a project that contains errors.

6.2 *Configuring the MicroEJ Framework*

The work-in-progress JPF can optionally include a properties file that defines the name of the fp file to load, overriding the selection of fp file made when the front panel was exported. This file is also used to specify some mockFP options (not mandatory).

To use this feature:

1. Create the file `frontpanel.options` in the work-in-progress JPF `mocks` folder.

2. Specify the fp file by setting the option `frontpanel.file`. For instance:

   ```
   frontpanel.file=STM32x0GEVAL.fp
   ```

3. The mockFP embeds some dynamic image decoders (see Images chapter). You can force these decoders to be disabled (by default the decoders are always enabled). Set the options `decoder.png.enabled` and `decoder.bmpm.enabled` to enable/disable the PNG decoder and the BITMAP monochrome decoder respectively. For instance:

   ```
   decoder.png.enabled=true

   decoder.bmpm.enabled=false
   ```

7 Designer Mode

Sometimes it is convenient to be able to design front panels using the Front Panel Designer without first creating a work-in-progress JPF. For example, a specialist graphic designer who neither has the skills or licence to create a JPF may want to work on the layout of a front panel.

The *Designer Mode* feature of the Front Panel Designer supports this way of working. By default, Designer Mode is disabled. It can be enabled using the **Window** → **Preferences** → **MicroEJ** → **Front Panel Designer** preference page options.

Once the mode has been enabled it is possible to create new Front Panel projects without associating them with a JPF. However, every Front Panel project must has access to a jar file that defines the set of widgets that can be displayed on the front panel. This jar file forms part of the JPF, but if no JPF is associated with the project then the path to the appropriate jar file must be specified. A default value for this path can be entered on the preference page to avoid having to enter it each time a project is created.

Designer Mode supports round-tripping of Front Panel projects between a designer (who is using Designer Mode) and a developer (using the normal mode). The designer can export the project once the basic layout is done, and the developer can import the project, associate it with a JPF (using the **Associated JPF** properties page), and add Java listener and display extension classes. If necessary the developer can export the updated project and pass it to the designer who can import it again without losing any project configuration. Equally, the designer and developer can share a project held in a source control system because the modal settings are specific to the user's MicroEJ workspace, not the project.

Note that if a project that has previously been associated with a JPF is used in Designer Mode the project may appear to have errors (red cross against the project root), but this will not affect the use of the project in Designer Mode.

8 Document History

Date	Revision	Description
20th June 2013	A	First release in this form - previously part of the UI Pack User Manual